Parsing

Phase	Input	output
Lexer	string of characters	string of tokens
Parser	string of tokens	Parse tree

Parser Roles

1- يأخذ (string of tokens) ويحولها لـ (Parse tree).

2- يميز بين (Valid and non valid string of tokens).

(CFG) Context free grammers

* Programming language constructs have recursive structure.
* Context-free grammers are natural notation for describing this recursive structure.

CFG consist of

1) set of terminals (T)

هي الشكل اللي عليه بمثل اللغة بتاعتي.

2) set of Non-terminals. (NT

3) A start symbol S (a non-terminal)

4) set of Production

ـه وظيفتها : تعمل (replace) في ال (string) الناتج.

$S \longrightarrow Ab$

$A \longrightarrow Bc$

$B \longrightarrow Cc$

$C \longrightarrow \varepsilon$

Terminals ـه b ، c

Non terminals ـه S, A, B, C

↳ that string doesn't accept except b's a or c's.

ـه وظيفة ال Grammer

١- يعرف كل ال (expressions) التي توصف اللغة الخاصة بي.

٢- يعمل إستبعاد أو إستثناء لكل ال (expressions) اللي مش معايا.

* How to Driven any Expression

1) إبدأ بـ (string) يحتوي الرمز "S" start symbol

2) استبدل ال Non-terminal وضع ما يناظره T or NT or ε

3) كرر رقم (2) حتى تصل لحالة عدم وجود Non terminal في ال (string)

* مقدرش أعمل (replacement) لأى (terminal)
بواسطة ال (Production tool).

* كل أما يوصل لـ (terminal) يبقى ثابت معانا.

*if you wanna to make derivation for
if if id then id else id fi then id else id fi

Expr ⟶ if Expr then Expr else Expr fi

⟶ if (if then Expr else Expr fi)
then Expr else Expr fi

*Which of strings are in language of given CFG

$S \longrightarrow aXa$

$X \longrightarrow \varepsilon$
$\quad | \; bY$

$Y \longrightarrow \varepsilon$
$\quad | \; cXc$

1) abcba $\Rightarrow$ XX

2) aCCa $\Rightarrow$ XX

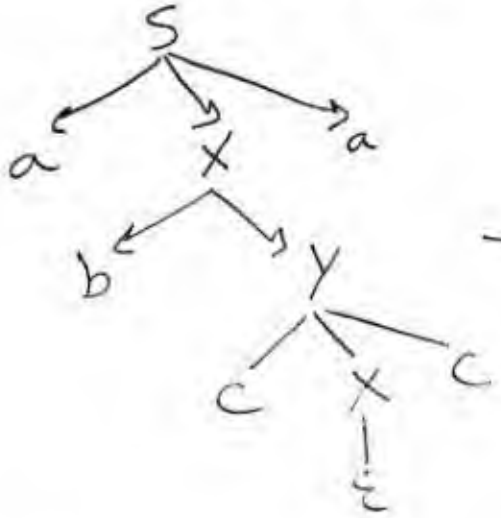
3) aba $\Rightarrow$ ✓✓

4) abcbcba ⟶ ✓✓

ـــ هنا خدهم نقطة نقطة

a) abcba

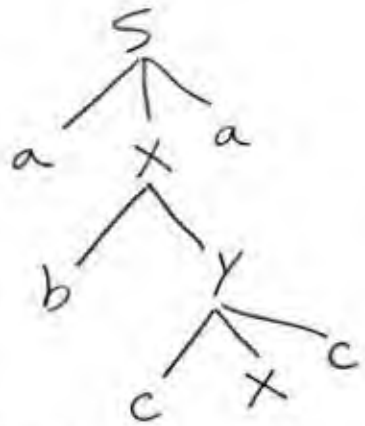
هنحاول نشوف الـ (CFG) اللي عندنا عاليمين في الهفحة السابقة بتتحقق مع النقطة a أم لا.



→ abcca لا تماثل

unaccepted

b) acca



"لا تحقق أيضا"

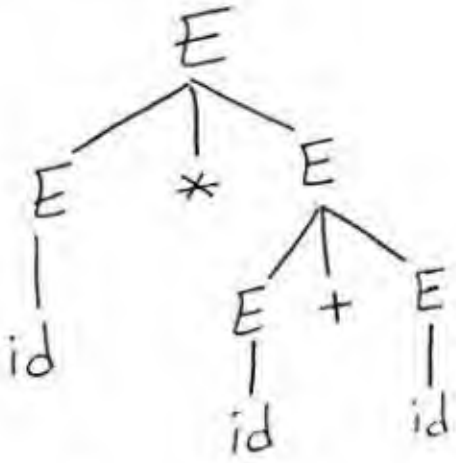
c) aba

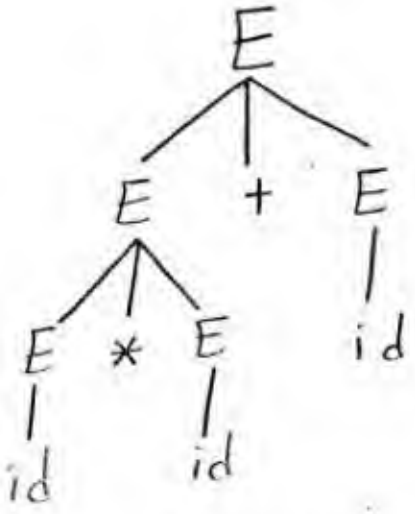
aba ⇒ accepted

[redacted]

4 [redacted]

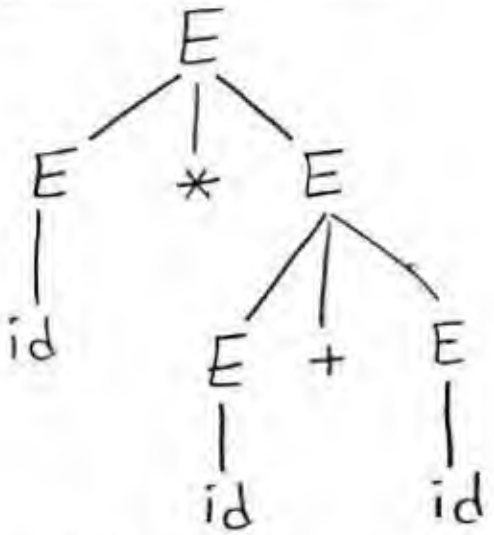
Ambigous



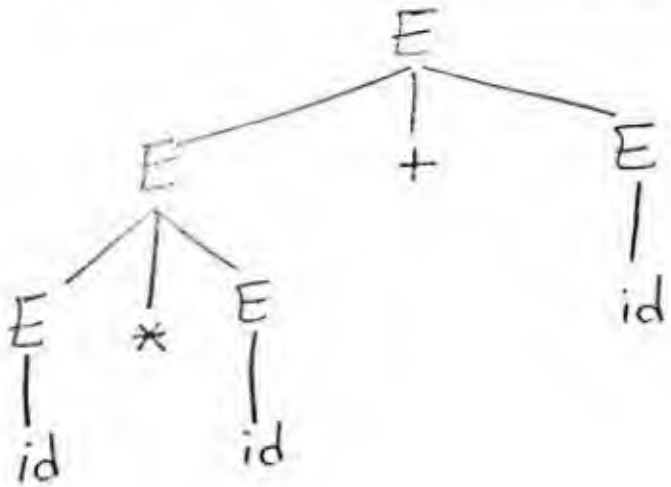


معناها إني وصلت بال (derivation) لنفس النتيجة
لكنه ب (2-Parsing trees) مختلفين.

Right most derivation



left most derivation



لنه تشتغل على ال (non terminal) في ال (right)، نفعله استبدال.

* أمثلة ال (Ambigous) متواجدة في ال (sheets).

* لا يوجد خطوات ثابتة كي تمنع وجود ال (Ambigous) بس الحل الأقرب إنك تعمل لل (rewrite) لل (grammer) بحيث ت (Avoid Ambiguity).

Ex

$S \rightarrow AaA \mid AbA$ ← (Ambigous) دي

$A \rightarrow c \mid s$

الحل ← تستبدل AbA بـ B وتكون

$S \rightarrow AaA \mid B$

~~$B \mid AbA$~~ $B \rightarrow AbA$

$A \rightarrow c \mid s$

The dangling else

~~MIF~~

→ else matches the closest then.

$S \rightarrow MIF$ (all then matched with an else)

$\mid UIF$ (some then is unmatched)

$MIF \Rightarrow$ if E then MIF else MIF

MIF: if-then else

MIF: if-then else

* if E then E else E

→ if E then UIF else MIF

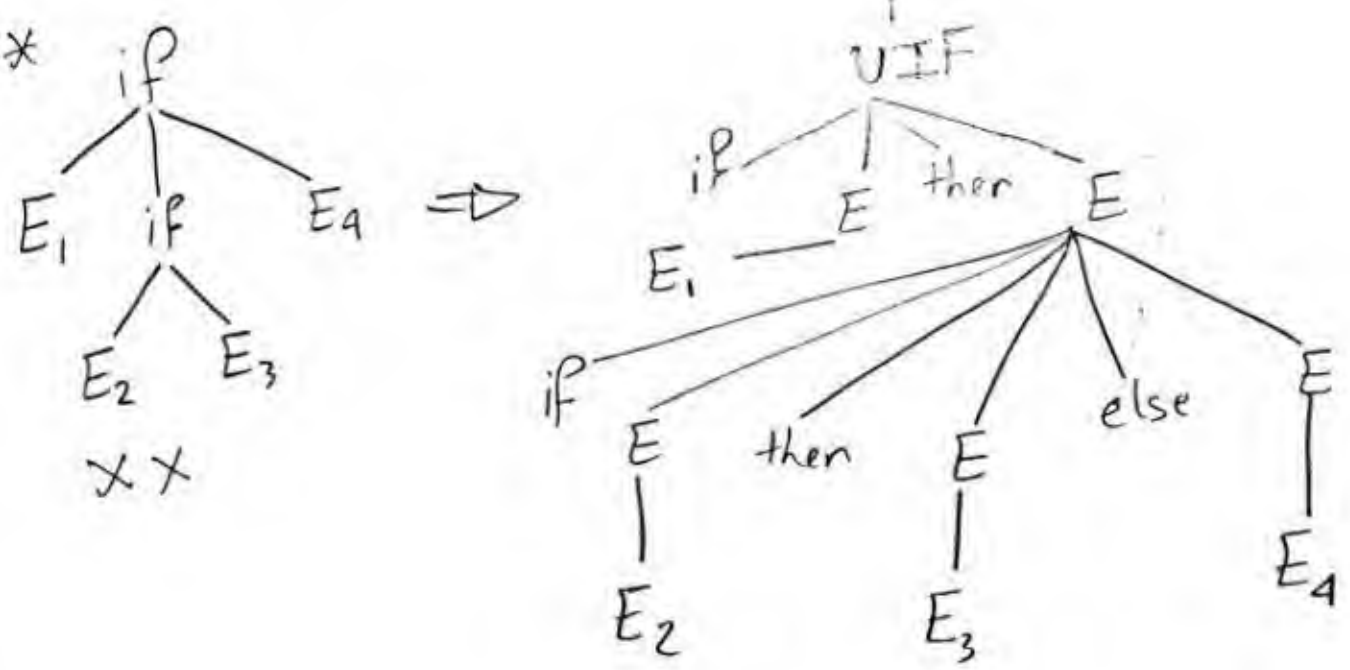
if then

لانعلم ← غير مقبول ← (if) تبع اى (else)

→ if E then MIF else UIF

if then else

accepted



- من أحد وظائف الـ (Compiler) إنه يعمل (handling) للـ (non-valid program or errors)

Panic mode 1

- يقوم بعمل تجاهل للـ (tokens) حتى يصل لنقطة واضحة تجعل الجملة صحيحة.

Ex

$(1 + + 2) + 3$

- من الشمال لليمين ١ ثم + ثم تتجاهل + القادمة حتى تصل لـ 2 ويكمل الجملة ويعطي رسالة إن فيه (syntax error) في مكان معين.

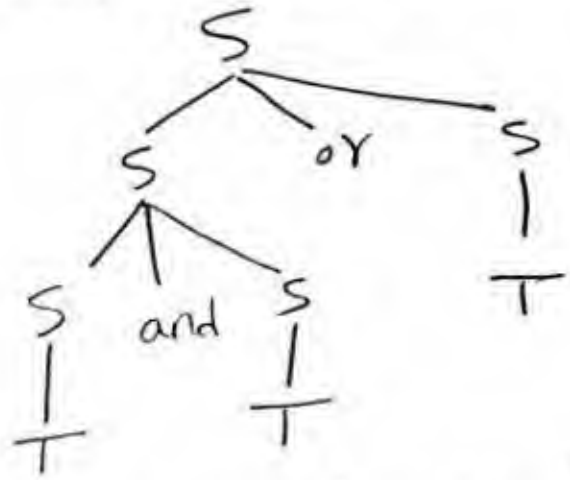
$1 + \xrightarrow{\text{حتى تصل}} 2$

- يتبقى (error Production)

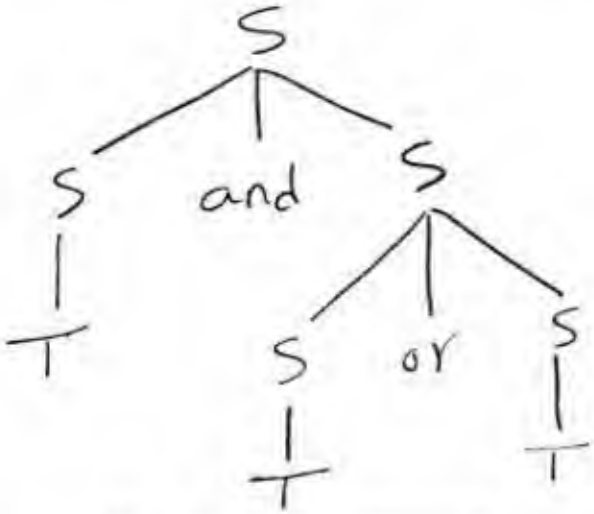
→ on Lecture 5 (slides) last 3-Pages.

* Presedence

← إيه اللى إتنفذ الأول



له (and) لها أولوية أعلى.

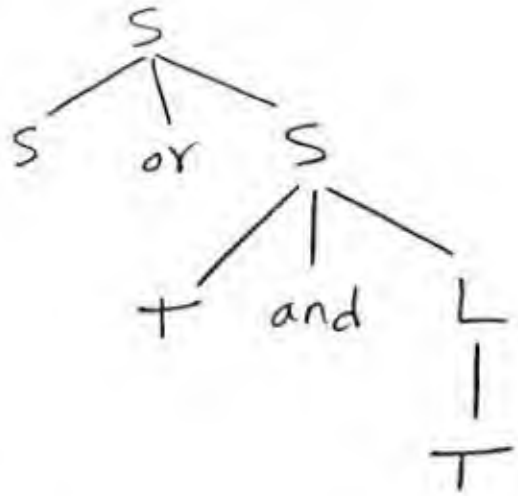


(or) لها أولوية أعلى.

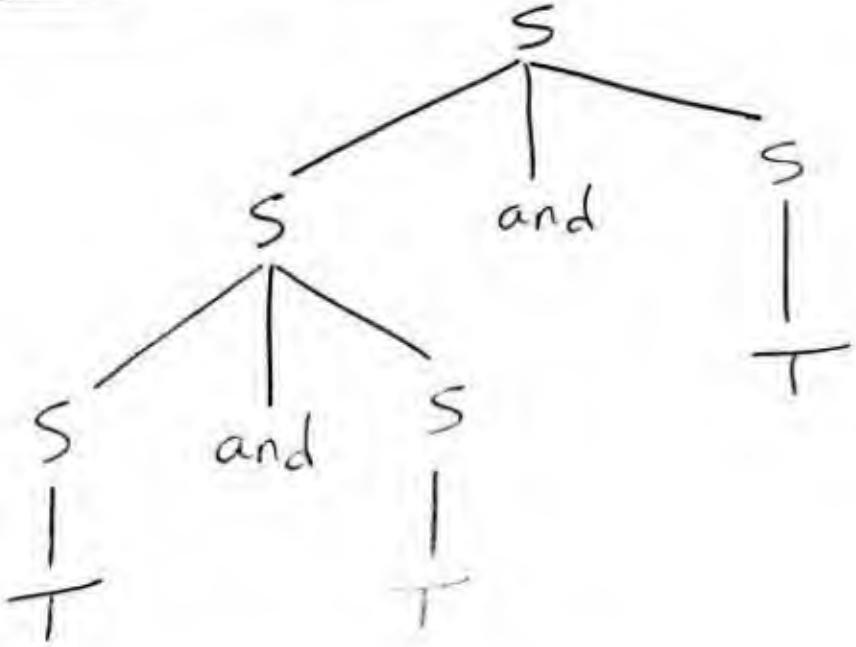
← لو عايز أجبره إنه يخلى (and) لها أولوية عن الـ (or)

S → S or S | L

L → true and L | true



* Left Associative



* Right Associative